Vstr

Daniel Plakosh, Software Engineering Institute [vita[1]]

2005-09-27 (updated 2008-07-17)

Vstr is a string library optimized to work with `readv()`/`writev()` for input/output. For example, you can `readv()` data to the end of the string and `writev()` data from the beginning of the string without allocating or moving memory. This also allows the library to work with data containing multiple zero bytes.

Development Context

String input and output

Technology Context

C, UNIX

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

C-style string input and output functions are prone to programmer mistakes that can result in buffer overflow vulnerabilities.

Description

Vstr is a string library optimized to work with `readv()`/`writev()` for input/output [Antill 06]. For example, you can `readv()` data to the end of the string and `writev()` data from the beginning of the string without allocating or moving memory. Vstr also works with data containing zero bytes.

Figure 1 shows a simple example of a program that uses Vstr to print out "Hello World." The library is initialized on line 8 of this example. The call to the `vstr_dup_cstr_buf()` function on line 9 creates a vstr from a C style string literal. The string is then output to the user using the `vstr_sc_write_fd ()` function on line 12. This call to the `vstr_sc_write_fd ()` function writes the contents of the s1 vstr to STDOUT. Lines 15 and 16 of the example are used to clean up allocated resources.

Unlike most string libraries, Vstr does not have an internal representation of the string that allows direct access from a character pointer. Instead, the internal representation is of multiple nodes, each containing a portion of the string data. This data representation model means that memory usage increases linearly as a string gets larger. Adding, substituting, or moving data anywhere in the string can be optimized to an O(1) algorithm.

Figure 1. Hello world using Vstr

```
1. #define VSTR_COMPILE_INCLUDE 1
2. #include <vstr.h>
3. #include <errno.h>
4. #include <err.h>
5. #include <unistd.h>
6. int main(void) {
7.   Vstr_base *s1 = NULL;
8.   if (!vstr_init()) err(EXIT_FAILURE, "init");
```

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/268-BSI.html (Plakosh, Daniel)

```
 9.   if (!(s1 = vstr_dup_cstr_buf(NULL, "Hello World\n")))
10.     err(EXIT_FAILURE, "Create string");
11.   while (s1->len)
12.     if (!vstr_sc_write_fd(s1, 1, s1->len, STDOUT_FILENO, NULL)) {
13.       if ((errno != EAGAIN) && (errno != EINTR)) err(EXIT_FAILURE, "write");
14.     }
15.   vstr_free_base(s1);
16.   vstr_exit();
17.   exit (EXIT_SUCCESS);
18. }
```

References

[ISO/IEC 99]	ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming languages — C*. International Organization for Standardization, 1999.
[Antill 06]	Antill, James. *Vstr documentation -- overview*[17], 2006.

Pearson Education, Inc. Copyright